Dynamische Neuplanung der Touren von Express Trucks unter Einbeziehung einer FCD-basierten Verkehrslage

Rüdiger Ebendt
Alexander Sohr
Louis Calvin Touko-Tcheumadjeu
Peter Wagner

Veröffentlicht in:
Multikonferenz Wirtschaftsinformatik 2012
Tagungsband der MKWI 2012
Hrsg.: Dirk Christian Mattfeld; Susanne Robra-Bissantz

Braunschweig: Institut für Wirtschaftsinformatik, 2012

Dynamische Neuplanung der Touren von Express Trucks unter Einbeziehung einer FCD-basierten Verkehrslage

Rüdiger Ebendt, Alexander Sohr, Louis Calvin Touko-Tcheumadjeu, Peter Wagner

Deutsches Zentrum für Luft- und Raumfahrt, Institut für Verkehrssystemtechnik,
12489 Berlin, Rutherfordstr. 2

E-Mail: {Ruediger.Ebendt|Alexander.Sohr|Louis.ToukoTcheumadjeu|Peter.Wagner}@dlr.de

Abstract

In den letzten zehn Jahren wurden im Deutschen Zentrum für Luft- und Raumfahrt (DLR) eine Reihe von prototypischen ITS-Diensten entwickelt, die auf „Floating Car Data“ (FCD) basieren. Eine Schlüsselanwendung ist dabei ein FCD-basiertes System zur Verkehrslenkung und Routenüberwachung. Dieses System wurde im BMWi-Förderprojekt „SmartTruck“ erweitert. Ein wichtiges Ziel dieses Projektes war die Verwendung von historischen und aktuellen Verkehrsinformationen für die energiesparende, optimierte offline-Planung und anschließende dynamische online-Neuplanung der Stoppreihenfolgen von Touren der DHL Express Trucks in Berlin. Diese Arbeit diskutiert die Architektur und eine wesentliche Neuerung des hierbei verwendeten FCD-Systems, eine Datenbank mit Performanzprofilen von Methoden zur Bearbeitung von Routenanfragen.

1 Einleitung

In den letzten Jahren haben Mobilitätsdienste an Bedeutung gewonnen, die auf sogenannten Floating Car Data (FCD)-Flotten basieren. Diese bilden eine Alternative zur lokalen Verkehrslagedetektion mit Induktionsschleifen, Infrarotsensoren und Videokameras. Das Prinzip besteht aus der Erfassung von im Verkehr „mitschwimmender” Fahrzeuge (Floating Cars). Dies geschieht durch die Übermittlung von Positionsmeldungen dieser Meldefahrzeuge an eine datenverarbeitende Zentrale. Die Positionen werden durch in den Fahrzeugen installierte GPS-Empfänger gewonnen. Der wesentliche Vorteil von FCD ist es, dass diese Daten für die Messung von Reisezeiten verwendet werden können. Diese sind eine begehrte Ressource für Telematik-, Logistik- und Routing-Anwendungen. Reisezeiten sind mit den vorgenannten konventionellen Methoden praktisch nicht erfassbar. FCD dagegen messen fortwährend und flächendeckend die durchschnittliche Reise-geschwindigkeit entlang vieler Streckenabschnitte.

Seit einem Jahrzehnt werden vom DLR FCD-Taxiflotten in mehreren europäischen Städten betrieben. Der Vorteil der Taxi-FCD-Technologie besteht in geringen Realisierungs- und Betriebskosten, da Taxis in der Regel bereits mit Digitalfunk und GPS ausgestattet sind. Zu den auf FCD basierenden Mobilitätsdiensten gehören neben der Verkehrslagedarstellung (siehe Bild 1) auch die auf Tagesganglinien basierende Verkehrsprognose, verkehrsabhängige LSA-Steuerungen und Rückstaulängenschätzung.

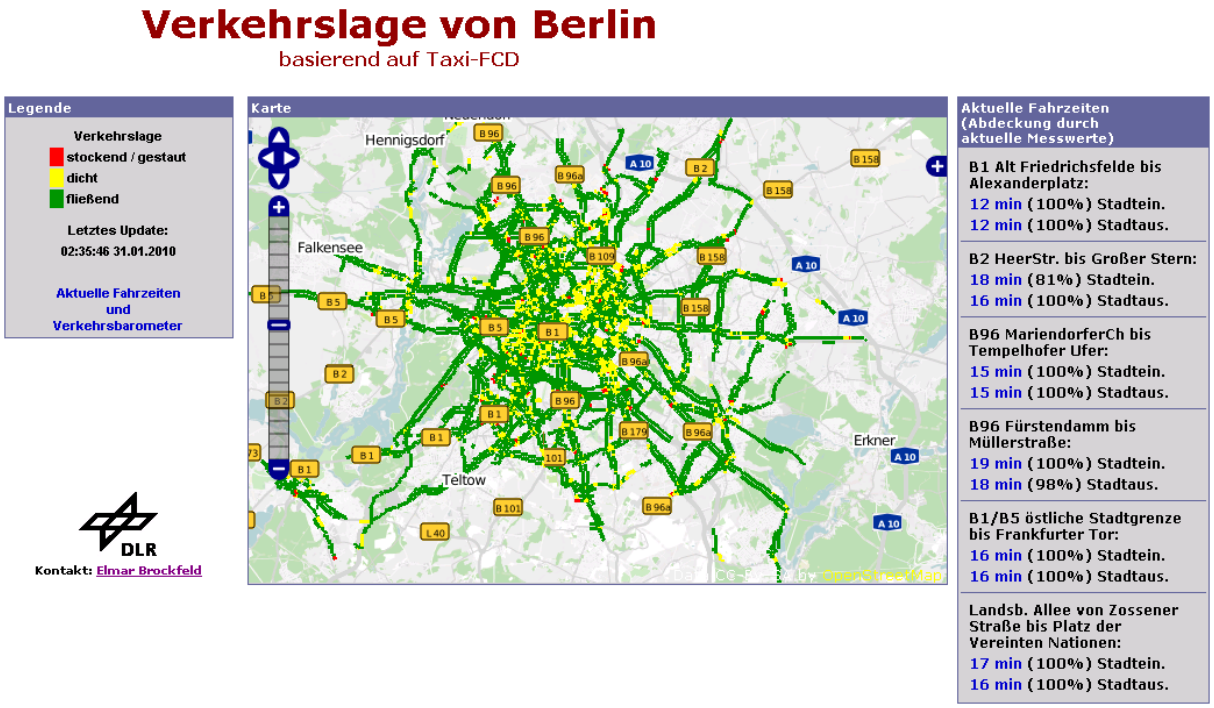

Bild 1: FCD-basierte Verkehrslage von Berlin (DLR)

Eine Kernapplikation des DLR ist ein Dienst zur Routenüberwachung, der auf historischen und aktuellen Reisezeiten für städtische Straßensegmente beruht, wie sie aus FCD gewonnen werden. Dieser Dienst wurde bis 2009 im Rahmen des BMWi-Förderprojektes „SmartTruck“ erweitert. Das Projekt „SmartTruck“ wurde von einem Konsortium realisiert, das aus DHL, einem Key-Player der Logistikindustrie, dem DLR und dem Deutschen Forschungszentrum für künstliche Intelligenz (DFKI) bestand. Ein wichtiges Ziel des Projektes war die Nutzung von historischen und aktuellen Verkehrsdaten für die energiesparende, optimierte offline-Planung und anschließende dynamische online-Neuplanung der Stoppreihenfolgen von Touren der DHL Express Trucks in Berlin.

Die klassische Transportplanung nach dem bisherigen Stand der Technik geht von einer optimalen Verkehrslage aus oder versucht diese (zum Teil nur unzulänglich) statisch durch in der Vergangenheit gewonnene Erfahrungswerte in der Transportplanung abzubilden. In der Realität bewegt sich aber die Fahrzeugflotte eines Transportunternehmens nicht durch einen störungsfreien Raum, vielmehr muss mit Verkehrsstörungen wie Staus, Baustellen und Unfällen gerechnet werden. Darüber hinaus können angekündigte Störungen wie Sperrungen von bestimmten Straßenabschnitten aufgrund von Großereignissen schon im Voraus mittelfristig bekannt sein.

Auf solche Störungen kann nach bisherigem Stand der Technik nicht oder nur unzureichend, keinesfalls jedoch präventiv reagiert werden. Ziel des Projektes „SmartTruck“ war es

deshalb, die Möglichkeiten einer effizienten Transportsteuerung mit zielgerichteten Eingriffen im Sinne einer Tourenplanung, die auf Verkehrsstörungen dynamisch reagiert, zu untersuchen. Voraussetzung hierfür ist ein ständiges Monitoring der (innerstädtischen) Verkehrslage.

Diese Arbeit gliedert sich wie folgt: In Abschnitt 2 wird eine kurze Übersicht über die Prozesse gegeben, die der bearbeiteten logistischen Fragestellung zugrunde liegen. Außerdem wird die Architektur des realisierten SmartTruck-Systems beschrieben. In Abschnitt 3 werden zunächst die Probleme beschrieben, die bei der Referenzierung der getrackten Positionsmeldungen von Fahrzeugen längs der Straßensegmente der digitalen Karte auftreten (also die beim sogenannten „Map-Matching“ auftretenden Probleme). Dann wird näher auf den im Taxi-FCD-System des DLR verfolgten routingbasierten Ansatz für das „Map-Matching“ eingegangen. Im Rahmen des Projektes „SmartTruck“ wurde ein neuartiger Ansatz zur Effizienzsteigerung bei der Bearbeitung der beim Map-Matching anfallenden Routing-Anfragen entwickelt. Dieser Ansatz ist Gegenstand der weiteren Ausführungen in Abschnitt 3. Der letzte Abschnitt schließt mit einer kurzen Zusammenfassung der erzielten Ergebnisse.

2 Prozesse und Systemarchitektur

Der Ausgangspunkt für den Transportplanungsprozess ist der Download der Abhol- und Lieferdaten für den aktuellen Tag. Die nächsten Schritte werden von Software-Komponenten durchgeführt, die von DHL Technologie-Partnern zur Verfügung gestellt werden: Dazu gehören die Geokodierung von Lieferadressen, die Übertragung der Geo-Positionen an die Routing-Server, und die Berechnung der Fahrzeiten zwischen etwa 3000 Adressen in Berlin. Dies wird durch Verwendung unterschiedlicher Fahrzeit-Matrizen für die jeweiligen Tageszeiten (und der damit erreichten Unterscheidung zwischen Haupt- und Schwachverkehrszeiten) erreicht. Die Basis für die Berechnung der Fahrzeiten sind historische Reisezeiten, die für jedes Straßensegment im DLR berechnet und in regelmäßigen Abständen aktualisiert werden. Die Fahrzeiten werden dann an die Planungs-Server der DHL Technologie-Partner für Abhol- und Lieferrouten übergeben. Sie dienen als Grundlage für den Optimierungsprozess, der in einer Zuordnung der Adressen an die Fahrzeuge innerhalb optimierter Stopp-Sequenzen resultiert.

Im Gegensatz zu bisherigen Planungsansätzen ist das System in der Lage die aktuelle Verkehrslage zu berücksichtigen. Dynamische Neuplanung passt den ursprünglichen Zeitplan an, um Strecken, die von Verkehrsstaus und identifizierten Baustellen betroffen sind, zu vermeiden. Die Neuplanung wird bei signifikanten Veränderungen der Verkehrslage eingeleitet: Im DLR laufen kontinuierlich Positionsmeldungen von über 4.000 Taxis in Berlin ein. Daraus werden aktuelle Reisezeiten für jedes Straßensegment berechnet und dem Traffic-Server gemeldet: Immer, wenn eine erhebliche Diskrepanz zu den jeweiligen historischen Reisezeiten (die bereits vorher vom DLR auf den Server übertragen wurden) erkannt wird, meldet der Traffic-Server dies dem Routenplanungs-Server. Gegebenenfalls initiiert dann der Routenplanungs-Server die Neuberechnung der Fahrzeit-Matrizen. In diesem Fall basiert die Neuberechnung auf den aktuellen Reisezeiten, wie sie gerade berichtet wurden, d.h. die aktuellen Reisezeiten überschreiben die historischen Reisezeiten. Bild 2 zeigt einen Überblick über die Architektur des SmartTruck Systems mit der skizzierten

dynamischen Tourenplanung und der „Dynamic Routing and Dispatch Application“ (DRADA) als Kern.

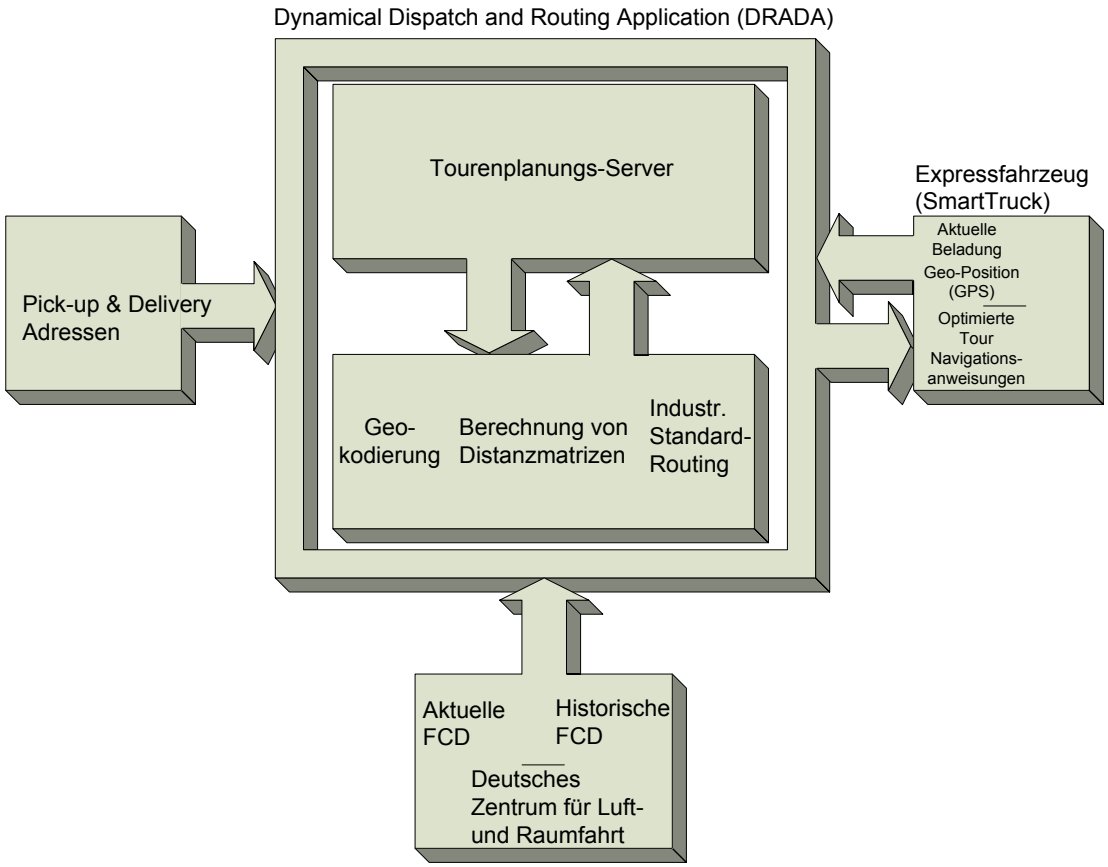

Bild 2: Architektur des SmartTruck-Systems

3 Map-Matching und Routing

Das „Map-Matching“-Problem besteht darin, den Zusammenhang zwischen Fahrzeug-Tracking-Daten (d.h. Trajektorien, also Sequenzen von Positionen mit Zeitstempeln) und dem Straßennetz herzustellen. Dabei können sowohl Stichprobenfehler als auch Messfehler auftreten. Das Problem des Stichprobenfehlers wird durch unbekannte Zwischensegmente, die während der Fahrt eines Messfahrzeugs zwischen zwei Positionsmeldungen passiert wurden, verursacht. Zu seiner Behandlung benutzt der hier beschriebene Ansatz einen Routing-Algorithmus. Dadurch wird auch sichergestellt, dass die berechnete Fahrzeugroute sich im Einklang mit topologischen und anderen Randbedingungen befindet, wie sie durch die Vernetzung der Straßenabschnitte gegeben sind. Zum Beispiel kann eine solche Route nicht plötzlich von einer Autobahn zu einer Ortsstraße führen, es sei denn, es bestünde eine geeignete Verbindung zwischen den beiden Netzsegmenten (in Form einer Ausfahrt), und es könnte aus diesem Grund eine entsprechende Route gefunden werden.

Der hier beschriebene Ansatz verbindet eine inkrementelle Vorwärtsstrategie mit einer globalen Rückwärtsstrategie. Zuerst wird in Fahrtrichtung eine inkrementelle Strategie angewendet, die Position um Position, und Kante um Kante bearbeitet: Eine Trajektorie wird durch eine (geordnete) Folge von Fahrten zwischen jeweils zwei Meldepositionen (Start- und Zielposition) beschrieben. Nun werden Lote auf alle Segmente, die innerhalb

einer bestimmten Entfernung (dem „Matching-Radius") zur Startposition eines aktuell bearbeiteten Paares von Positionsmeldungen liegen, gefällt. Die Segmente innerhalb des Matching-Radius´ bezeichnen wir im Folgenden als Kandidatensegmente. Jedes Kandidatensegment wird am Lotfußpunkt in zwei Hilfssegmente aufgeteilt, wobei am Lotfußpunkt ein zusätzlicher Hilfsknoten eingerichtet wird. Jeder zusätzlich eingerichtete Hilfsknoten dient als Anfangs- bzw. Endpunkt des ausgehenden und des eingehenden Hilfssegmentes. Er dient außerdem als möglicher Startpunkt einer Zwischenroute für die Fahrt zwischen den zwei gemeldeten Positionen (d.h. jeder solcher Hilfsknoten ist ein potentieller "Startknoten"). Dann wird das gleiche für die zweite gemeldete Position getan. Die hieraus resultierenden zusätzlichen Hilfsknoten sind potentielle Endpunkte der gesuchten Zwischenroute (d.h. jeder solche Hilfsknoten ist ein potentieller „Endknoten").

Dann wird für die beiden Mengen der potentiellen Start- und Endknoten eine „Multi-Source“-Variante des Dijkstra-Algorithmus´ [2] verwendet, um für die aktuell betrachtete Fahrt zwischen zwei Meldungen eine Reihe von potentiellen Routen zu bestimmen. Die Zielsetzung hierbei ist es, eine beste (oder kürzeste) Route von einem Knoten der ersten Menge zu einem Knoten der zweiten Menge zu finden.

Zu beachten ist, dass es sich dabei nicht um eine Verallgemeinerung des „Single Pair Shortest Path“ (SPSP)-Problems handelt. Das beschriebene Problem kann nämlich wieder zu einem SPSP-Problem reduziert werden[1]. Die Klasse der SPSP-Probleme wird offenbar von der Klasse der „Single Source Shortest Path“ (SSSP)-Probleme subsumiert, und daher kann das vorliegende Problem nach nur leichter Modifikation von jedem der für die Lösung des SSSP-Problems zur Verfügung stehenden Verfahren gelöst werden.

Daher wird nacheinander eine schnelle Methode zur Lösung des SSSP-Problems für die aufeinanderfolgenden Fahrten zwischen jeweils zwei Meldepositionen aufgerufen: Die Zielposition der vorherigen Fahrt wird die Startposition des nächsten Durchgangs, und die Endknoten werden die Startknoten des nächsten Durchgangs. Die bereits ermittelte beste Route hin zu diesen Knoten wurde im vorherigen Durchgang gespeichert. Die Kosten dieser Route werden an den nächsten Durchgang weitergeleitet, indem sie als Anfangskosten an den neuen Startknoten annotiert werden. Dadurch ist gewährleistet, dass der Routing-Prozess insgesamt eine Suche nach einem kostenminimalen Pfad für die gesamte Trajektorie verfolgt.

Dieser kostenminimale Pfad wird in einer zweiten Phase des Verfahrens bestimmt. Dabei wird eine globale Strategie umgesetzt, die mögliche Routen für die aufeinanderfolgenden Fahrten zwischen jeweils zwei Meldepositionen rückwärts zu einer global optimalen Route für die gesamte Trajektorie zusammensetzt. Statt einer „Look-Ahead“-Strategie fester Tiefe oder eines in der Anzahl der Schritte begrenzten Backtrackings, um die Qualität der verschiedenen möglichen Pfade auszuwerten (siehe z. B. [8]), minimiert die Methode über

[1]Um dies einzusehen, nehmen wir an, dass zwei neue Knoten *s*, *t* zum Graphen hinzugefügt werden, wobei *s* Vorgängerknoten von allen Knoten der ersten Menge ist, und *t* Nachfolgerknoten aller Knoten der zweiten Menge. Die hierbei neu eingefügten Kanten erhalten Kosten (bzw. eine Länge) von 0. Wenn wir nun diesen erweiterten Graphen betrachten, kann aus jedem besten (kürzesten) Weg von *s* nach *t* leicht ein bester (kürzester) Weg von einem Knoten der ersten Menge zu einem Knoten der zweiten Menge im ursprünglichen Graphen gewonnen werden (und damit wird bereits das ursprüngliche Problem gelöst).

alle möglichen Pfade im Straßennetz, die mit einer gegebenen Trajektorie übereinstimmen. Nach Konstruktion genügt es dabei, einfach die kostenminimalen Pfade rückwärts zu verketten: Beginnend beim am besten bewerteten Endknoten der letzten Fahrtroute, wird die Rückwärtsverkettung iteriert, bis sie an einem der potentiellen Startknoten der ersten Route zwischen den beiden ersten Meldepositionen der Trajektorie endet.

Zur Behandlung des Problems von Messfehlern können verschiedene Qualitäts- und Fehlermaße in die Kostenfunktion mit aufgenommen werden (z. B. die schwache Fréchet-Distanz, s. [1]). Um den Stichprobenfehler zu behandeln, zielt die Kostenfunktion auch auf kürzeste oder schnellste Wege zwischen den Positionsmeldungen ab. Der Routing-Algorithmus nutzt auch historische Reisegeschwindigkeiten, die offline durch Map-Matching von historischen Tracking-Daten gewonnen wurden. Beim online Map-Matching entwickelt sich die Trajektorie erst nach und nach in Echtzeit und ist nicht sofort verfügbar. Daher wird eine kleine Anzahl *k* von kürzlich eingetroffenen Positionsmeldungen und die entsprechenden, noch unbestätigten „Matches“ (also Zuordnungen zu Positionen auf Segmenten des Straßennetzes) in einem Puffer vorgehalten. Sie können verwendet werden, um zu entscheiden, ob ein früherer „Match“ basierend auf den neu angekommenen Positionsmeldungen verändert werden sollte.

Dadurch können vorläufig auf die Karte referenzierte Tracking-Daten bereits an andere Teile der FCD-Verarbeitungskette weitergeleitet werden, bevor die Trajektorie vollständig vorliegt, und die entstehende Verzögerung beträgt nur *k* der jeweils für einen Verarbeitungsschritt nötigen Zeitintervalle. Bei online-Anwendungen, bei denen ein erstes Ergebnis schnell vorliegen soll und dessen Qualität auch nachträglich noch verbessert werden kann, kann die Verwendung auch ohne jede Verzögerung erfolgen.

Zudem werden alle Daten (inklusive der berechneten Ergebnisse) im Hinblick auf verrauschte GPS-Signale und unplausible Werte in den Datenquellen gefiltert. Dies beinhaltet den Ausschluss von nicht-repräsentativen Daten. Diese treten auf, wenn Taxis spezielle Busspuren benutzen dürfen oder wenn die Taxis Passagiere aufnehmen oder absetzen (siehe auch [8]).

3.1 Dispatcher-Komponente

Aufgrund der Echtzeit-Bedingungen, die eine Verkehrslagedarstellung und die damit verbundenen Mobilitätsdienste erfüllen müssen, muss das Map-Matching online erfolgen.

Damit muss der verwendete Ansatz zur Lösung des SSSP-Problems so performant wie möglich sein. Dies gilt insbesondere vor dem Hintergrund eines FCD-Servers, der große Mengen an eingehenden Positionsdaten in kurzer Zeit prozessieren muss.

Das DLR startete in Berlin mit mehreren hundert Taxis und hat dieses erste Berliner System mittlerweile zu einer großen Flotte von mehr als 4.000 Taxis ausgebaut.

Schnelle Methoden zur Lösung des SSSP-Problems können neben der Performanzsteigerung bei online-Anwendungen auch zum offline-Fall beitragen: In einem Zeitraum von mehreren Jahren wurden im DLR mehr als eine Milliarde Positionen von Taxi-Flotten gesammelt. Angesichts dieser Datenmengen können auch die Laufzeiten von offline-Prozessierungen dieser historischen Daten durch performantere Methoden zur Lösung des SSSP deutlich gesenkt werden.

Der Ansatz, der im Rahmen des Projektes „SmartTruck“ entwickelt wurde, integriert mehrere Varianten von Dijkstra [2] bzw. A* [4] in einem Algorithmus. Eine Kernkomponente schätzt hier zunächst ab, wie “schwer” es sein wird, eine gegebene Routenanfrage zu beantworten. Dann wird die eingehende neue Position an diejenige Variante überreicht, die aufgrund früherer Erfahrungen am besten geeignet erscheint.

- Dies ist oft die Methode mit der kleinsten erwarteten Laufzeit. Dabei wird der folgende Umstand ausgenutzt: Es gibt Methoden, die einerseits sehr performant arbeiten wenn Anfragen für längere Distanzen verarbeitet werden müssen (z.B. eine Anfrage, die vom Benutzer eines webbasierten Routenplanungsservice stammt), die aber andererseits nur mäßig bis schlecht für kürzere Entfernungen geeignet sind (der Grund hierfür ist ein hoher initialer Overhead dieser Methoden). Für andere Methoden wiederum ist es genau umgekehrt.
- Andernfalls wird eine Methode gewählt, die zu der aktuellen Anforderung an die Qualität der Lösung passt. Beispielweise kann es für eine online-Applikation vorteilhaft sein, eine erste Antwort schnell zu liefern – auch wenn dies zunächst mit nur moderater Qualität geschieht. Dies gilt insbesondere dann, wenn es möglich ist, diese erste Antwort später zu verbessern. Im Gegensatz hierzu haben offline-Anwendungen normalerweise keine solchen strikten Echtzeitanforderungen. Hier gibt es also keine starken Gründe, die Qualität z.B. während der offline-Evaluation von Verkehrsdaten zu reduzieren.

Die Ergebnisse von im Folgenden noch näher beschriebenen Experimenten resultierten in einer Datenbank mit Performanzprofilen. Diese Datenbank enthält beispielsweise Informationen darüber, welche Variante von Dijkstra oder A* bei welchem Luftlinienabstand zweier aufeinanderfolgenden Positionsmeldungen die beste Performanz aufweist. Wenn der Dispatcher nun auf eine bestimmte solche Luftliniendistanz trifft, wählt er die entsprechende Methode aus, um das aktuelle Routing-Problem zu lösen. Da auch die anderen Kriterien des Dispatchers von ähnlicher Einfachheit sind, ist der entstehende zusätzliche Overhead durch diese Dispatcher-Entscheidungen vernachlässigbar.

Der Dispatcher berücksichtigt außerdem den erforderlichen Grad der Qualität der Lösung. Wenn z.B. die Luftliniendistanzen aufgrund einer hohen Meldefrequenz klein sind, haben die Meldefahrzeuge in der Regel bei der Fahrt von Start- zu Zielposition keine oder nur wenige Straßensegmente überfahren. In diesem Fall wird der Dispatcher die verfügbaren approximativen A*-Varianten [5][7][10] bevorzugen, da sie die benötigte Route sehr schnell liefern und dies in den meisten Fällen mit keinem bzw. mit einem nur sehr geringen Verlust an Qualität einhergeht.

Insgesamt ergeben sich die folgenden verwendeten Kriterien:

- Luftliniendistanz (aufeinanderfolgender Positionsmeldungen)
- Segmenthierarchie (z.B. befinden sich Startpunkt und Endpunkt auf oder in unmittelbarer Nähe einer Stadtautobahn)
- AOI („Area of Interest“), also: Befinden sich Start und Ziel innerhalb einer bestimmten „Bounding Box“, die ein bestimmtes Gebiet einer Stadt beschreibt (der „AOI“), für welches frühere Erfahrungen bereits gesonderte Performanzprofile lieferten
- Anwendungsspezifische Qualitätsanforderungen
- Anwendungsspezifische Laufzeitanforderungen
- Benutzerspezifische Informationen

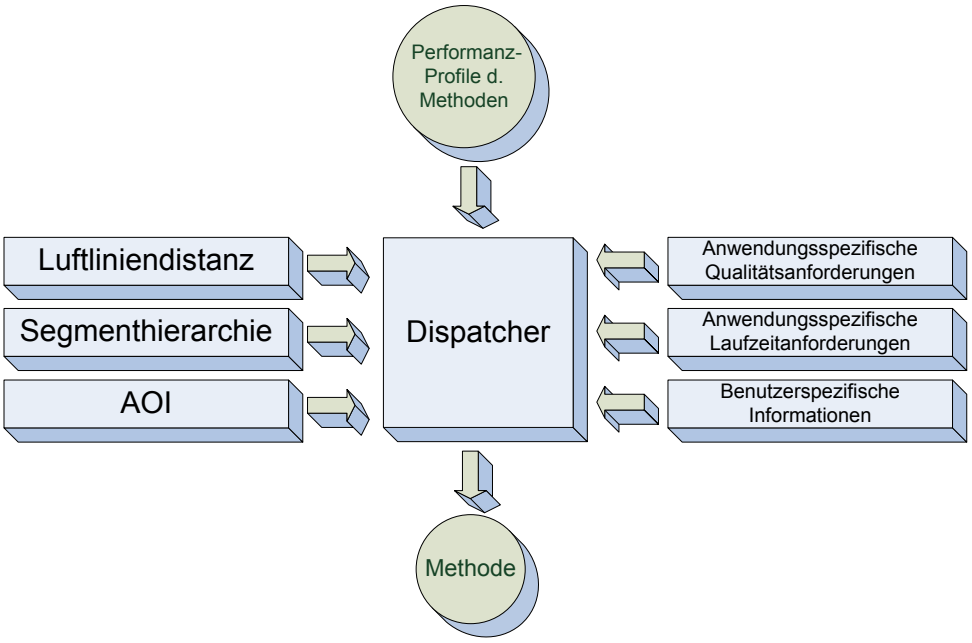

Bild 5: Funktionsweise der Dispatcher-Komponente

Die Länge der durchschnittlichen Luftliniendistanz hängt von der Meldefrequenz ab. Diese Frequenzen können dabei unter Umständen je nach betrachteter Stadt und Flotte erheblich variieren: Z. B. bekam das System zu Projektbeginn für die DHL Express-Trucks in Berlin (sowie für bestimmte Taxi-Flotten in Hamburg) durchschnittlich etwa alle 30 Sekunden eine Positionsmeldung, während es bei der damaligen Berliner Taxi-Flotte nur etwa eine Positionsmeldung alle zwei Minuten war.

Der Dispatcher beschleunigt den Routing-Prozess, ohne dass dies mit einem merklichen Verlust in der Qualität der Referenzierung verbunden wäre: Für die Hamburger Taxi-Flotte, hat die Methode mit dem größten durchschnittlichen Qualitätsverlust (eine neuartige Integration von Pohl´s gewichtetem A* [7] in den Algorithmus A_ε von Ghallab und Allard [10]), einen durchschnittlichen Qualitätsverlust von weniger als vier Promille. Die verfügbaren Varianten unterscheiden sich im verwendeten Basis-Algorithmus (Dijkstra-Algorithmus [2], A* [4], gewichtetes A* [7] oder die erwähnte neue Methode) oder in der Umsetzung der

Prioritätswarteschlange für sogenannte „offene“, d.h. noch unbearbeitete Knoten. Letztere kann mit verketteten Listen oder auf Basis eines Fibonacci-Heaps realisiert werden [3].

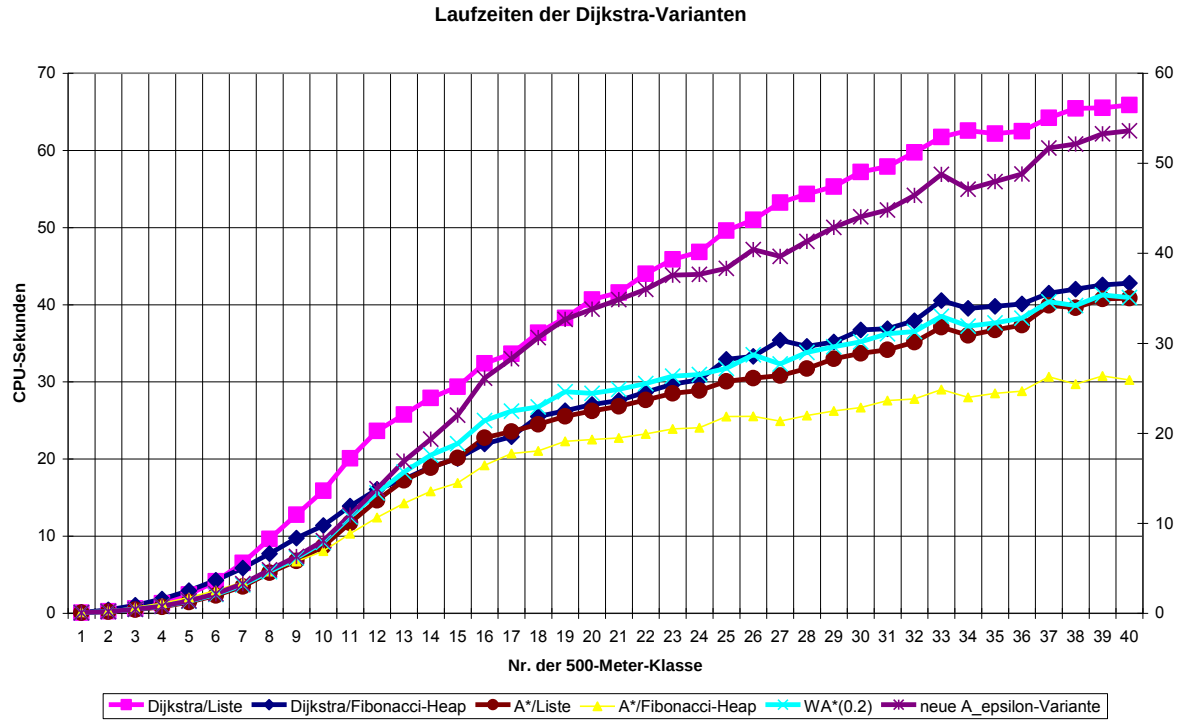

Bild 3: Laufzeiten der Dijkstra-Varianten

Bild 3 zeigt die durchschnittliche Laufzeit[2] der jeweiligen Dijkstra-Varianten, die auf zufällig generierten Quelle-Ziel-Paaren verschiedener Distanzen in Hamburg angewendet wurden. Die Paare wurden bezüglich ihrer Luftliniendistanz in Klassen eingeteilt. Dabei wurde eine neue Klasse für jeden neuen 500-Meter-Schritt eingerichtet, und zwar derart, dass 10.000 Paare in jeder Klasse von 0 bis 20 km enthalten waren. Betrachtet man die längsten Distanzen, übertrifft die Performanz der Varianten mit Implementierungen der Prioritätswarteschlange, die auf einem Fibonacci-Heap basieren, die derjenigen Verfahren mit entsprechenden listenbasierten Implementierungen. Weiterhin übertrifft die Performanz des A*-Verfahrens deutlich die des Dijkstra-Verfahrens. Gewichtetes A* (bei Anwendung eines Gewichts von $\varepsilon = 1{,}2$) ist trotz seiner listenbasierten Implementierung der Prioritätswarteschlange sehr schnell, und listenbasiertes A_ε, ist, obschon es schneller als listenbasiertes Dijkstra ist, nicht schneller als die anderen Varianten. Insgesamt ergibt sich gerade das erwartete Verhalten, und so gesehen können die Ergebnisse für die längeren Distanzen kaum überraschen.

Der Nutzen der listenbasierten Methoden zeigt sich erst beim Blick auf die *kürzeren* Strecken. Wie Bild 4 für Entfernungen von 0 bis 2 km zeigt, sind die listenbasierten Verfahren deutlich schneller, da der Fibonacci-Heap bei der Initialisierungsphase hohe Overhead-Kosten aufweist. Die modifizierte Variante von A_ε hat in etwa dieselbe Performanz wie gewichtetes A*.

[2] Die Experimente wurden auf einer Maschine mit einem Intel® Core™2 Duo CPU, betrieben bei 2,4 GHz ausgeführt.

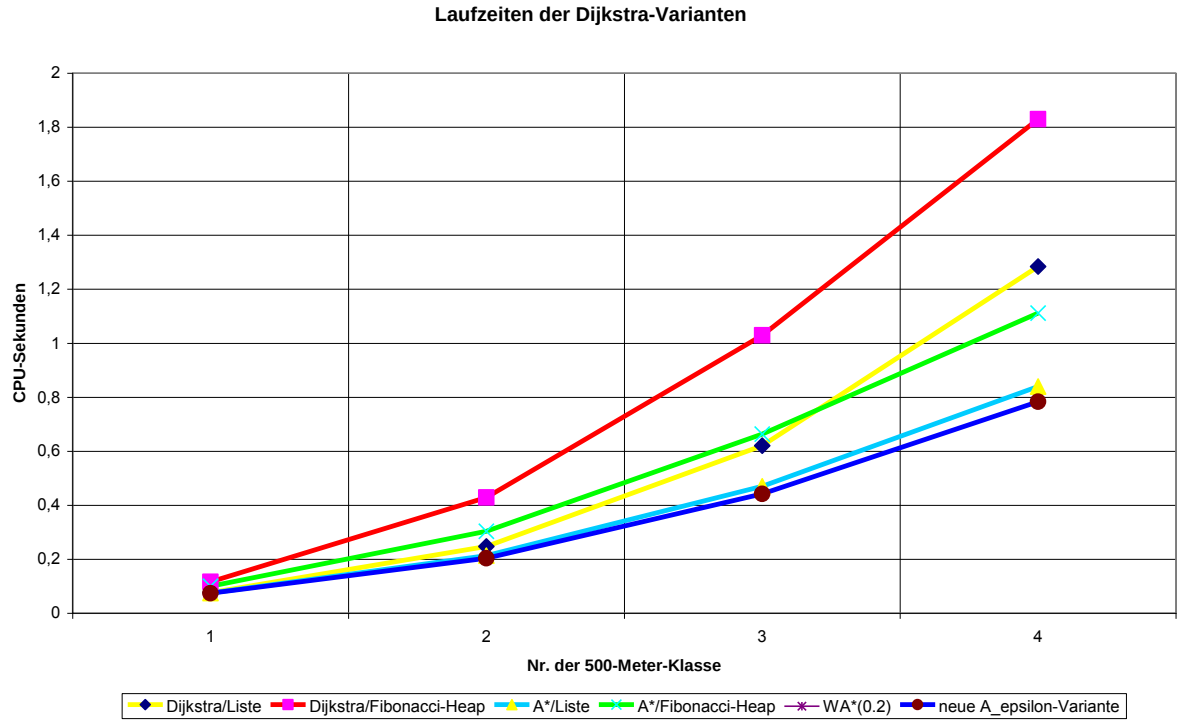

Bild 4: Laufzeiten von mehreren Dijkstra-Varianten für kleinere Distanzen

A_ε führt eine Tiefensuche durch, solange gewährleistet ist, dass die resultierenden Kosten der Lösung um nicht mehr als einen Faktor $1+\varepsilon$ von den Kosten der Minimallösung abweichen (ε wurde in den Experimenten auf 0,2 gesetzt). Diese Strategie hat sich für die Arbeit mit kurzen Distanzen als die effektivste Variante erwiesen.

Der Trend, der sich in diesem Experiment mit rein synthetischen Quelle-Ziel-Paaren gezeigt hat, konnte auch in der Realität beobachtet werden: Bedingt durch die unterschiedlichen Meldefrequenzen in Berlin und Hamburg, war im Projekt „SmartTruck“ A* als Variante mit einem Fibonacci-Heap (wobei Pohl´s gewichtete Variante verwendet wurde, sobald der geringe Qualitätsverlust zu vernachlässigen ist) das Verfahren, das in Berlin am häufigsten aufgerufen wurde, während in Hamburg die modifizierte A_ε-Methode selbst gewichtetes A* an Performanz übertraf und die kleinsten Laufzeiten lieferte. In Berlin reduziert die beschriebene Dispatcher-Strategie die Laufzeit der innerhalb des Map-Matchers aufgerufenen Routing-Anfragen um ca. 30%.

3.2 Qualität der mit dem Dispatcher-Ansatz berechneten Routen

Bild 2 zeigt exemplarisch, wie sich der prozentuale Qualitätsverlust für die Verfahren Dijkstra, A* und für ausgewählte Varianten derselben mit steigendem Luftlinienabstand entwickelt (aus Platzgründen kann nicht jedes Dispatcherkriterium und nicht jedes der Verfahren im Methodenpool des DLR hier behandelt werden).

Der Qualitätsverlust von gewichtetem A* (mit ε=0,2) bleibt auch für die längeren Distanzen unterhalb von 0,1%. Damit liefert die Methode in der Praxis weitaus bessere Resultate als durch die Theorie der ε-Zulässigkeit garantiert wird. Auch für andere Problemfelder sind ähnlich positive Erfahrungen berichtet wurden (s. [5]).

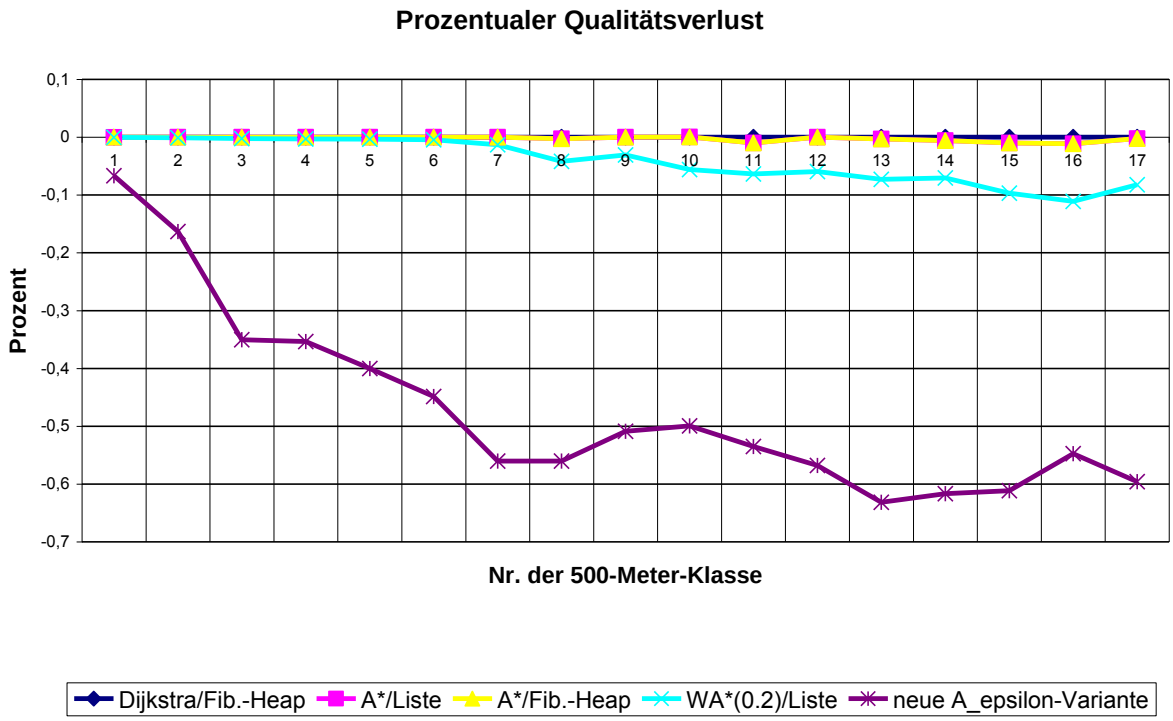

Bild 5: Prozentualer Qualitätsverlust für ausgewählte Dijkstra-Varianten

Da die heuristische Funktion, wie sie innerhalb des exakten A*-Verfahrens verwendet wird, bestimmten Pragmatismen unterliegt[3], können die so berechneten Routen eine sehr geringe Abweichung von der jeweils optimalen Route aufweisen – diese Abweichung ist jedoch für die Praxis vernachlässigbar. Die Integration von gewichtetem A* in A_ε ist diejenige Methode, die mit dem größten Qualitätsverlust behaftet ist. Allerdings bleibt selbst bei dieser Methode der Qualitätsverlust unterhalb von 0,5%, und dies auch bei den niedrigsten Meldefrequenzen innerhalb der Taxi-Flotten des DLR bzw. bei den daraus resultierenden Luftlinienabständen zwischen zwei aufeinanderfolgenden Meldepositionen.

Außerdem wählt der Dispatcher die A_ε-Modifikation bevorzugt bei hohen Meldefrequenzen: Diese Methode ist vor allem für Routing-Probleme über kürzere Distanzen vorteilhaft, da es sich um eine schnelle, an Tiefensuche orientierte Variante von A* handelt. Hier resultiert die Verwendung der neuen A_ε-Modifikation in Laufzeitreduktionen von 5 bis 10%, während mit der herkömmlichen A^*_ε-Methode [6] keine klare Verbesserung erzielt werden konnte.

Insgesamt ergeben sich so die bereits erwähnten ca. 30% durchschnittliche Laufzeit-reduktionen für die DLR-Flotten mit niedrigeren Meldefrequenzen, während für Flotten mit höheren Meldefrequenzen immer noch Beschleunigungen von 5 bis 10% erzielt werden können. Dies ist möglich ohne dass der damit einhergehende Verlust der Qualität der berechneten Routen einige wenige Promille übersteigt.

[3] Als heuristische Funktion wird die Reisezeit verwendet, die aus der bei einer angenommenen maximalen Fahrzeuggeschwindigkeit zurückgelegten Luftliniendistanz resultieren würde. Eine konservative Strategie würde diese Maximalgeschwindigkeit so hoch ansetzen, dass die Zulässigkeit der heuristischen Funktion sichergestellt ist. Hier wird allerdings eine etwas aggressivere Strategie verwendet, da dies die Effektivität der heuristischen Funktion signifikant verbessert.

4 Schlussfolgerung

In dieser Arbeit wurden die Architektur und grundlegende Methoden eines Systems zur Transportplanung einer Flotte von Express Trucks beschrieben. Das System wird durch Floating Car Data (FCD) von Taxis und Trucks mit Echtzeit-Verkehrsinformationen versorgt. Die daraus abgeleiteten historischen und aktuellen Reisezeiten für Straßensegmente ermöglichen dem System, Störungen zu erkennen und auf die veränderte Verkehrssituation durch die Berechnung einer optimierten, energiesparenden Alternativroute zu reagieren. Eine neuartige „Dispatcher-Komponente“ ermittelt mit Hilfe einer Datenbank von Performanzprofilen zu jeder während des „Map-Matchings“ auftretenden Routenanfrage die jeweils geeignetste Methode. Dies führte zu durchschnittlichen Performanzsteigerungen von 5 bis 30%.

5 Literatur

[1] Alt, H; Efrat, A; Rote, G; Wenk, C (2003): Matching planar maps. Journal of Algorithms 49, 262-283.

[2] Dijkstra, EW (1959): A note on two problems in connexion with graphs. Numerische Mathematik 1, 269-271.

[3] Fredman, ML; Tarjan, RE (1987): Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34, 596-615.

[4] Hart, P; Nilsson, N; Raphael, B (1968): A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 2, 100-107.

[5] Korf, RE (1993): Linear-space best-first search. Artificial Intelligence 62(1), 41-78.

[6] Pearl, J; Kim, J (1982): Studies in semi-admissible heuristics. IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-4(4), 392-399.

[7] Pohl, I (1970): Heuristic search viewed as path finding in a graph. Artificial Intelligence 1 (3), 193-204.

[8] Brockfeld, E; Wagner, P; Passfeld, B (2007): Validating travel times calculated on the basis of taxi floating car data with test drives. In: *Proceedings of the ITS World Congress,* Beijing, China.

[9] Chawathe, SS (2007): Segment-based map matching. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, Istanbul, Türkei, 13-15.

[10] Ghallab, M; Allard, DG (1983): A_ε: An efficient near admissible heuristic search algorithm. In: *Proc. Int. Joint Conf. on Artificial Intelligence*, Karlsruhe, Deutschland, 789-791.